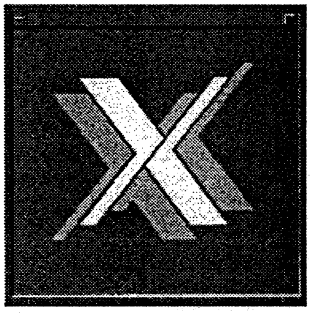
SourceGate® II

for

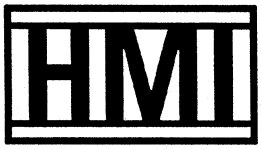
HMI Background Mode Debuggers (BMDs)



User's Guide Addendum

Version 2.0.0

January 30, 1996



Huntsville Microsystems, Inc. © 1996

The contents of this document are believed to be accurate. However, Huntsville Microsystems, Inc. assumes no responsibility for the use of this document and issues no warranties, implied or otherwise, on the contents herein. Furthermore, Huntsville Microsystems, Inc. reserves the right to revise this publication and to make changes to the contents, without obligation to notify any persons of such revisions or changes.

Copyright © 1996 by Huntsville Microsystems, Inc.

SourceGate is a registered trademark of Huntsville Microsystems, Inc.
SUN logo and OpenWindows are trademarks of Sun Microsystems, Inc.
SPARC is a trademark of SPARC International Inc.
Pentium is a registered trademark of Intel Corporation.
Microsoft is a registered trademark and Windows is a trademark of Microsoft Corporation.

This document was created entirely with AmiPro, which is a trademark of Lotus Development Corporation.

TABLE OF CONTENTS

New Features **1**

Overview 1

Description Of New Features 2

C++ Class Browser Window 2

C++ Class Sub Browser Window 3

Preferences Window 4

Register Map Window 5

Symbols In Assembly and Tracking Assembly Windows 6

New BMD Commands 7

S_PMBA (Set Programmable Module Base Address) Command 7

S_ROMWIDTH (Set ROM Width) Command 8

Changes In Operation From Earlier Versions **13**

Single-Step Window No Longer Supported 13

File/Change Directory Function No Longer Supported 13

Changes In Watch/ Watch In Real Time Window Operation 13

Adding a Watch Variable 14

Entering Watch Variables Manually 14

Watching A Symbol 15

Watch Variable Type 15

Watch Variable Overrides 16

Watch Variable Address 16

Watch Variable Display Type 16

Watch Entry Window 16

Watch Entry Window Buttons 17

Watch Entry Type Selection Window 18

Watch Entry Type Selection Window Buttons 18

Watch Entry Override Selection Window 19

Watch Entry Overrides Selection Window Buttons 19

Watch Entry Format Selection Window 19

Deleting a Watch Variable 20

TABLE OF CONTENTS (Cont.)

Saving a Watch Window 20
Restoring a Saved Watch Window 20
Closing a Watch Window 20
Modifying Watch Window Variable Values 20
Watch Window Buttons 21
Changes In Memory Window Operation 22
Memory Window Buttons 22
Memory Compare Window 23
Memory Compare Window Buttons 24
Memory Fill Window 24
Memory Fill Window Buttons 24
Memory Goto Window 25
Memory Goto Window Buttons 25
Memory I/O Window 25
Memory I/O Window Buttons 26
Memory Move Window 26
Memory Move Window Buttons 26
Memory Search Window 27
Memory Search Window Buttons 27
Memory Test Window 28
Memory Test Window Buttons 28
Memory Display Format Window 28
Memory Display Format Window Buttons 29
Memory Output Window 29
Memory Output Window Buttons 30
General Information 31
Problem Fixes 31
Known Problems 33
Known Problems Affecting All Platforms 33
Known Problems Affecting The Windows Platform 33
Known Problems Affecting The SUN/SPARC Platform 34
Known Problems Affecting The HP Platform 35

1 New Features

Overview

Several new features have been added to this release of SourceGate II for HMI Background Mode Debuggers (BMDs). Some of the more prominent new features are:

C++ Class Browser/Sub-Browser Windows
C++ Class Browser and Sub-Browser windows have been added to allow you to browse all C++ classes in your C++ source file.

Preferences Window
This window allows the user to define preferences for certain SourceGate II default values.

Register Map Window
This window allows the user to define the value used for a register when register variables are used by the compiler.

Symbols in VASM and Tracking Assembly Windows
The VASM and Tracking Assembly windows now show symbols.

Added Support For Windows 95
This release of SourceGate II will run correctly under Windows 95.

ELF/DWARF Support
SourceGate II now can read the ELF/DWARF file format.

Memory Window Can Now Be Minimized
The Memory window now has a minimize button.

BMD File Downloads Have Been Sped Up
File downloads when using the BMD are now 6 times faster in most cases. This should eliminate users having to wait long periods of time for large files to load.

New S_PMBA (Set Programmable Module Base Address) Command *(Not Available For The MPC505)*
The command S_PMBA allows the programmable module base address to be viewed and set.

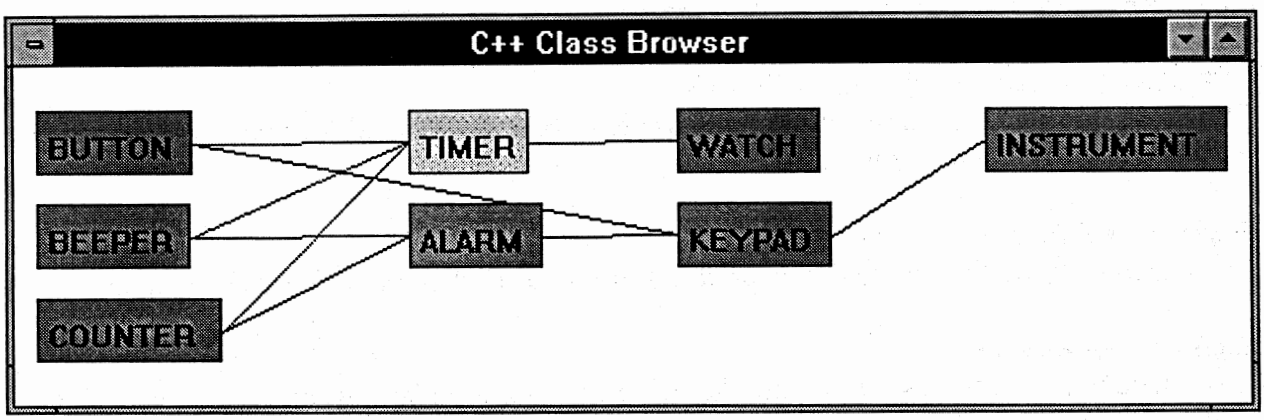
New S_ROMWIDTH (Set ROM Width) Command *(68360 Only)*
The command S_ROMWIDTH allows the ROM width to be viewed and set.

More detailed information on several of the above new features is provided in the following sections.

Description Of New Features

More detailed information on several of the new features found in this release of SourceGate II is provided in the following sections.

C++ Class Browser Window

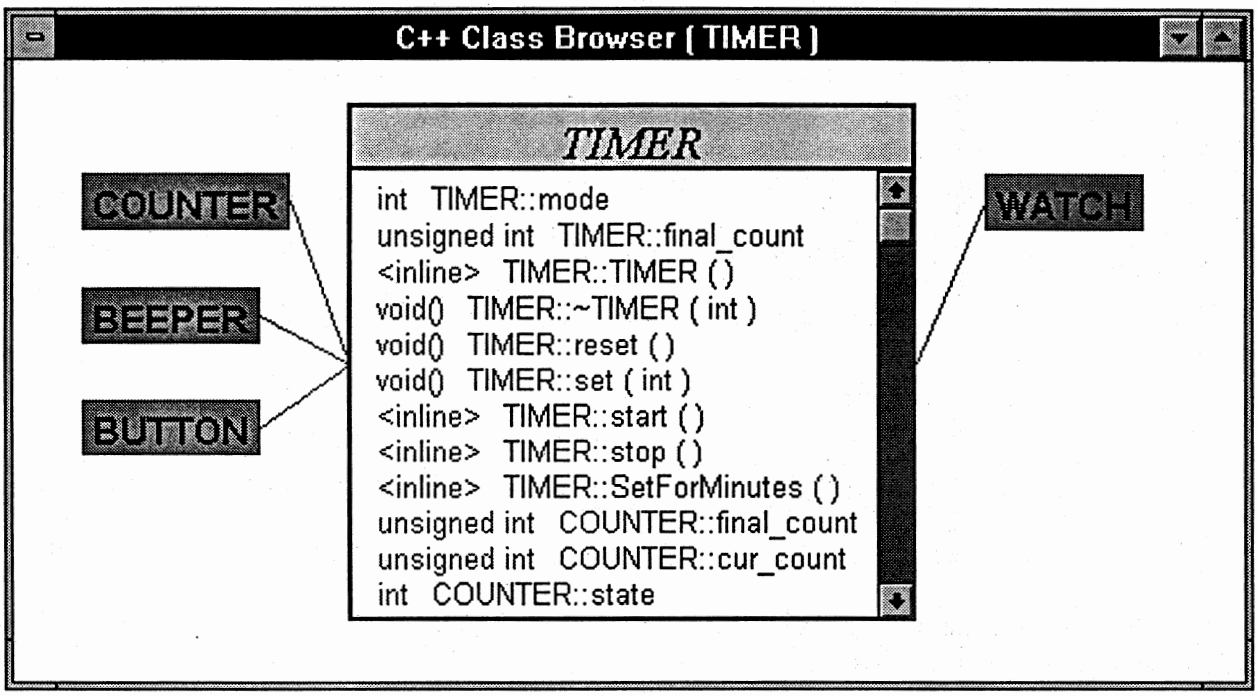


This window allows you to browse all C++ classes contained in the source file. The Base (or Parent) classes are shown on the left side of the window. Derived (or Child) classes are shown to the right of the Base classes. Also shown are the lines of inheritance between the classes. These lines show you what classes were derived from the Base classes.

Class objects are shown in green. To select a class object, click on it once. Its color will then change from green to yellow. If you have selected a Base class, you will notice that the paths of inheritance to all classes derived from the Base class are shown in red. If you have selected a Derived class, you will notice that the paths of inheritance from all related Base classes will be shown in red.

When you double-click on a class object, a C++ Class Sub Browser Window is shown for that class object. This Sub Browser window shows paths of inheritance along with all elements of its own class and all elements of the classes that it was derived from (Base/Parent).

C++ Class Sub Browser Window



This window allows you to browse all C++ classes contained in the selected class object. This Sub Browser window shows paths of inheritance along with all elements of its own class and all elements of the classes that it was derived from (Base/Parent). It also shows any Base and Derived classes that are associated with the selected class. You can double-click on these other classes while in the sub browser window and this will pull up sub windows for those classes.

In addition, this window will show all data members and member functions. A member function will contain bracket () and a data member will not contain brackets. Some examples may help.

Data Member Example:

unsigned int Timer::final_count

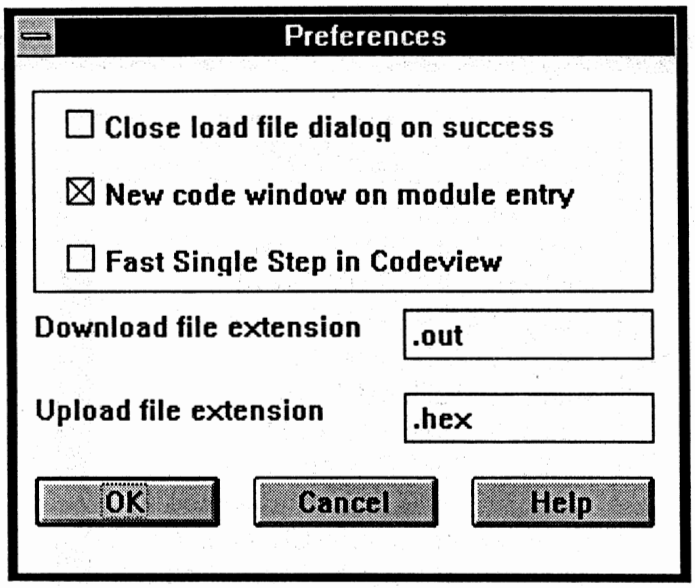
In the above example, "unsigned int" is the type of element (shown in red), "Timer" is the class (shown in blue), and "final_count" is the variable that is a member of "Timer" (also shown in blue).

Member Function Example:

<inline> Timer::start(int)

In the above example, "<inline>" is the type of element (shown in red), "Timer" is the class (shown in blue), and "start" is the Member function that also shows the type of argument for the function ("int" in this example). Both the brackets and the argument types that they contain will be shown in red.

Preferences Window



This window allows the user to define how SourceGate II behaves in certain areas. Each of the fields shown in this window are outlined below.

Close Load File Dialog On Success:
This option, when checked, will close the file load status window upon successful completion of the file load process. If this option is not checked, the file load status window will remain on the screen until the Close button is pressed.

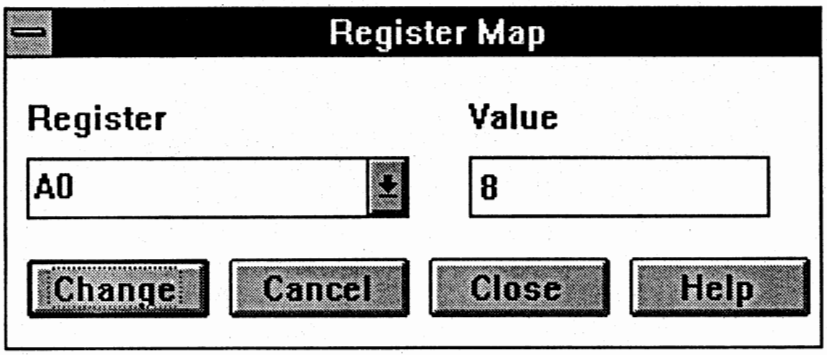
New Code Window On Module Entry:
This option, when checked, will open a new Code window when the Code window steps into another module, leaving each of these new windows on the screen. If this option is not checked, stepping into another module will close the current module's Code window and will then open the new module's Code window. This prevents multiple Code windows from being on the screen at the same time.

Fast Single Step In Codeview:
This option, when checked, will allow faster single stepping in the Code window. However, not all emulators support this feature reliably. If you have trouble single stepping reliably when this option is checked, disable it and try again.

Download File Extension:
This field determines the file extent that is to be used when downloading a file using the 'File/Download Hex File' command.

Upload File Extension:
This field determines the file extent that is to be used when uploading a file using the 'File/Upload Hex File' command.

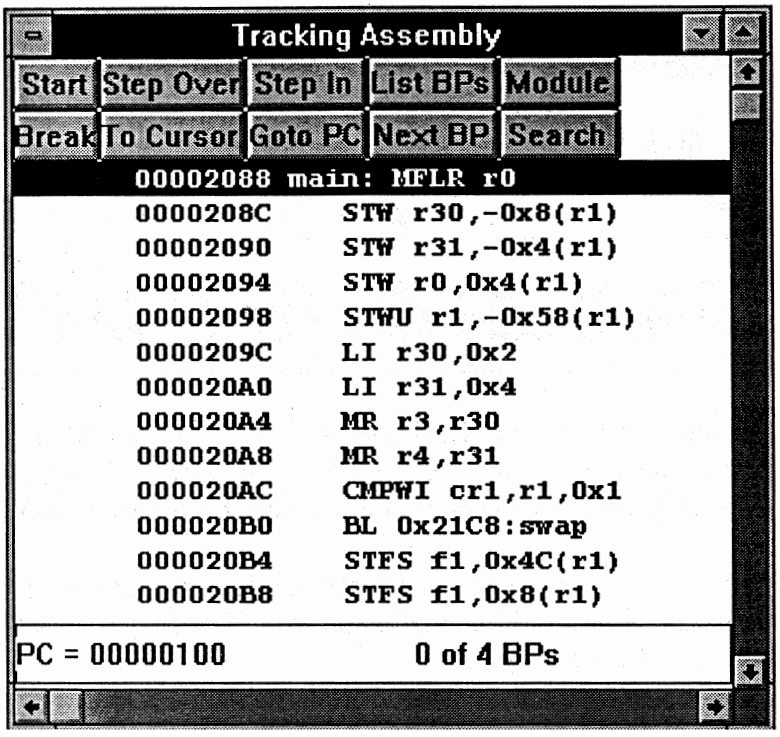
Register Map Window



This window allows the user to define the value used for a register when register variables are used. This value should match the value assigned to the register by the compiler. Once you have entered the desired value, press the Change button to store the change in SourceGate II.

Consult the register variables section of your compiler documentation for information on what value to set for a particular register. Note that some compilers may not use register variables. In that case, this window will be meaningless.

Symbols In Assembly and Tracking Assembly Windows



Symbols are now shown in the VASM, Assembly, and Tracking Assembly windows. This feature was added in response to customer requests.

New BMD Commands

A description of several new BMD commands follows.

S_PMBA (Set Programmable Module Base Address) Command *(Not Available For The MPC505)*

A new command has been added to SourceGate II BMD. The command S_PMBA (which may be entered using the command window or in a command script) allows the programmable module base address to be viewed and set. The format of the S_PMBA command is:

S_PMBA [<New Base>]

where:

New Base: is the desired location of the module base address

If no base address is specified, the current value is displayed. Otherwise, the specified address is validated, and the SourceGate II BMD sets the new address as follows:

68340/349/360:
The MBAR register is programmed to move the peripheral modules to the specified base address. The EXR command reads the MBAR register on every access to determine the register addresses.

68331/332:
The base address is recorded, and the EXR command uses this recorded address to determine register addresses.

Note: The SourceGate II BMD does not attempt to program the base address or determine the base address dynamically because the bit which is used to select the base address is itself relocatable, thus making it impossible for the SourceGate II BMD to determine the base address reliably. Therefore, for these processors, the S_PMBA command is used only to notify the SourceGate II BMD of the base address to use.

Some examples may help clarify how this command functions:

To determine the current base address:

```
S_PMBA
Peripheral Module Base Address = FFFFF000
```

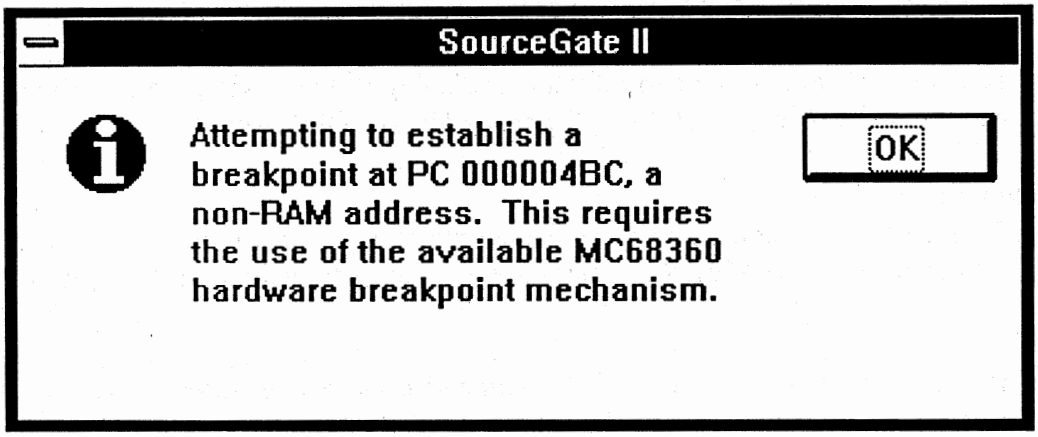
To set the base address to a new value:

```
S_PMBA FFF000
Peripheral Module Base Address = 00FFF000
```

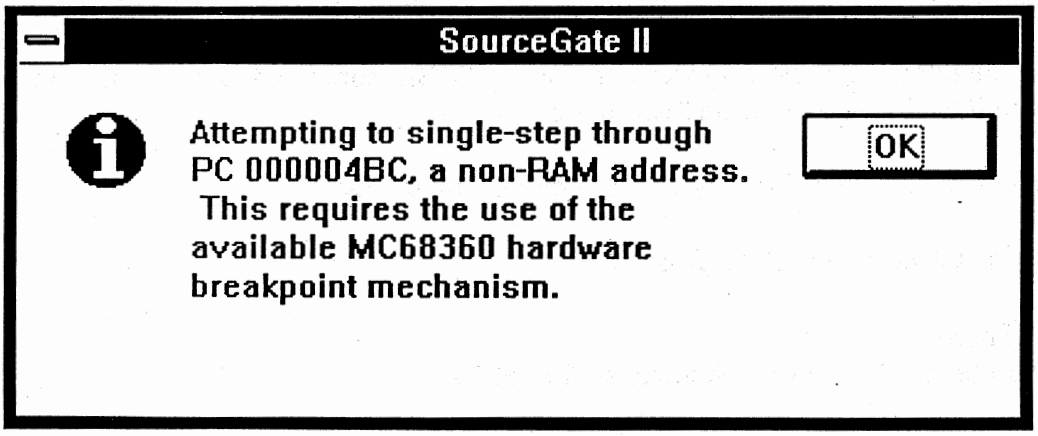
S_ROMWIDTH (Set ROM Width) Command *(68360 Only)*

The 68360 processor contains registers which may be used to establish a breakpoint in ROM. SourceGate II BMD for the 68360 uses these registers when the user tries to set a breakpoint in, or single-step through, ROM.

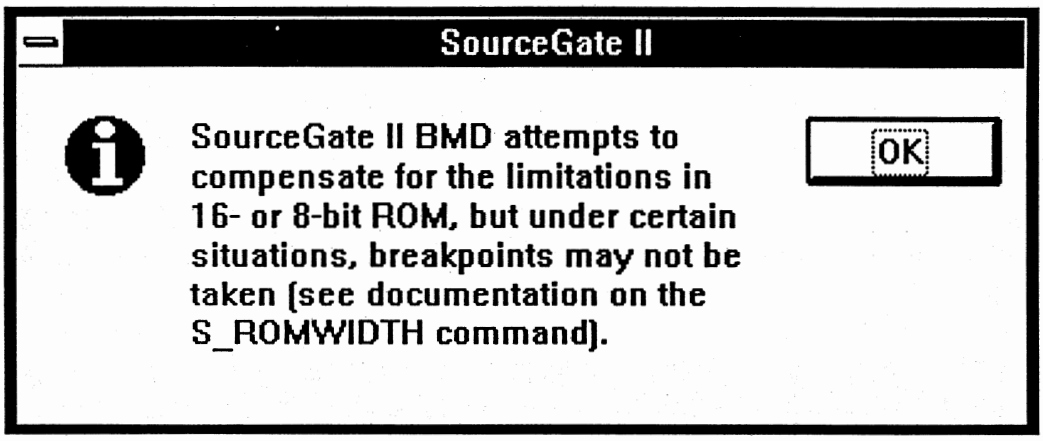
These registers do not work in all cases. If the PC lies in 16 bit-wide or 8 bit-wide memory, programming the registers works only if the Break Address Register (BKAR) is set to a 32 bit-aligned address. If it is set to a 16 bit-aligned address, the processor does not stop executing. Thus, when the user first attempts to set a breakpoint in ROM, SourceGate II displays the following series of dialogs:

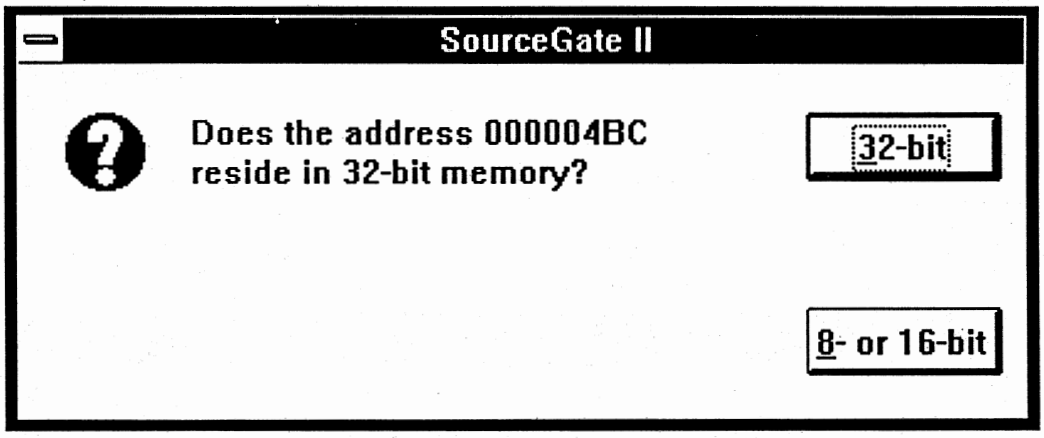

SourceGate II
Attempting to establish a breakpoint at PC 000004BC, a non-RAM address. This requires the use of the available MC68360 hardware breakpoint mechanism.
OK

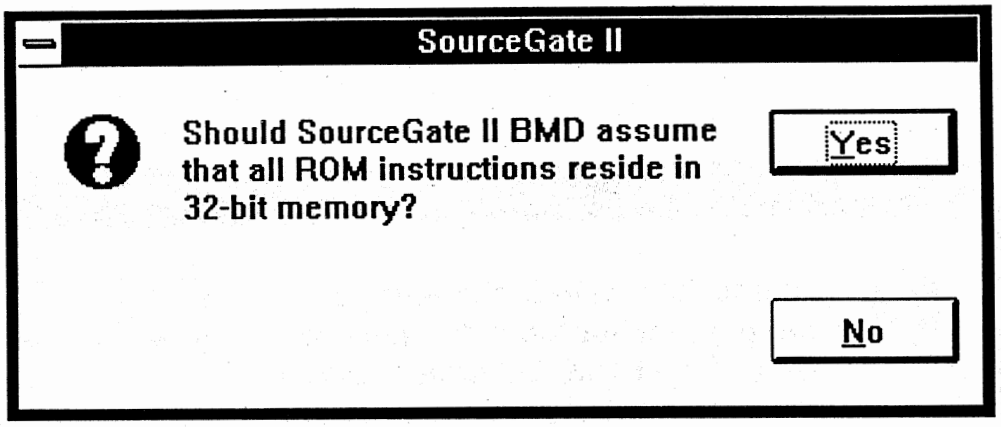
OR


SourceGate II
Attempting to single-step through PC 000004BC, a non-RAM address. This requires the use of the available MC68360 hardware breakpoint mechanism.
OK

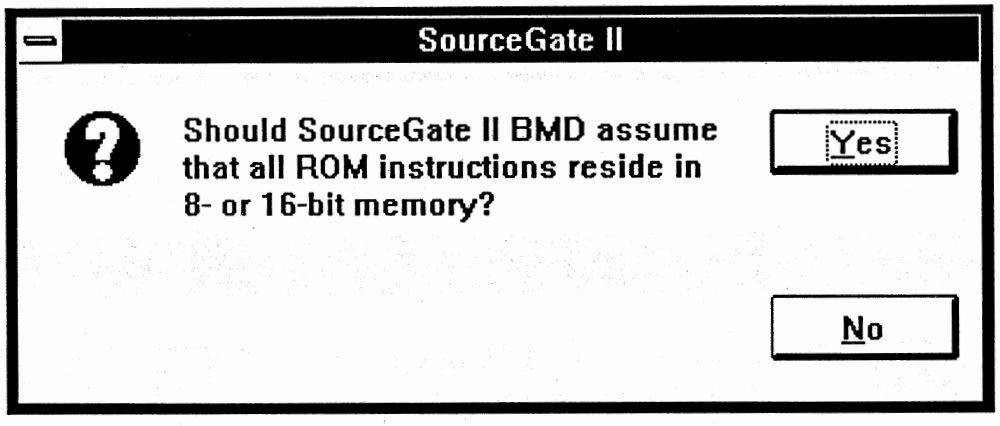
The dialog shown will depend on the operation being performed (single-stepping or setting a breakpoint).


SourceGate II
SourceGate II BMD attempts to compensate for the limitations in 16- or 8-bit ROM, but under certain situations, breakpoints may not be taken (see documentation on the S_ROMWIDTH command).
OK

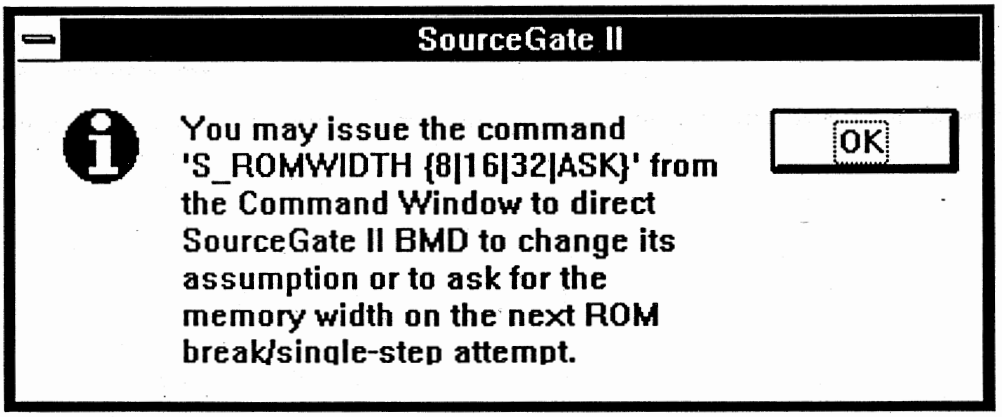

SourceGate II
Does the address 000004BC reside in 32-bit memory?
32-bit
8- or 16-bit



OR

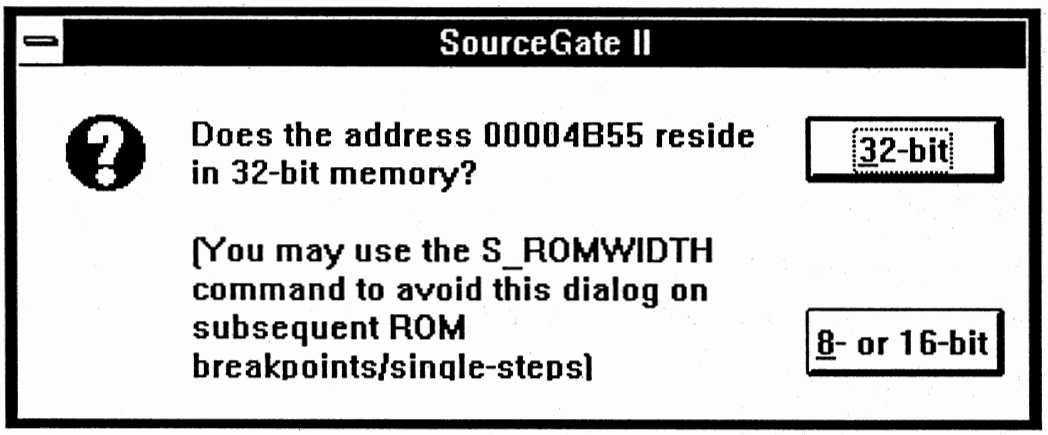


The dialog shown will depend on the answer given for the previous dialog.


SourceGate II
You may issue the command 'S_ROMWIDTH {8|16|32|ASK}' from the Command Window to direct SourceGate II BMD to change its assumption or to ask for the memory width on the next ROM break/single-step attempt.
OK

On subsequent attempts to set breakpoints, if the user has specified (via the above dialog boxes or by using the S_ROMWIDTH command) that SourceGate II should always assume a specified size, SourceGate II does assume the specified size, and doesn't ask any questions.

Otherwise, the next time the user tries to establish a breakpoint in ROM or single-step while in ROM, they will get this dialog:


SourceGate II
Does the address 00004B55 reside in 32-bit memory?
[You may use the S_ROMWIDTH command to avoid this dialog on subsequent ROM breakpoints/single-steps]
32-bit
8- or 16-bit

At this point, if the user specifies a 32-bit ROM, then the breakpoint address register is simply set to the address specified. **Note: Make sure that, if "32-bit ROM" is selected, the system actually does have a 32-bit ROM. If "32-bit ROM" is selected but the breakpoint address is 16-bit aligned, the breakpoint will never occur, even if the instruction is executed.**

If the user specifies a 16-bit ROM, then SourceGate II rounds up the address to the next 32-bit boundary. This usually means that the breakpoint will occur a little later than normal. However, if the instruction where the breakpoint is specified is a 16-bit branch instruction which is 16-bit aligned, then the breakpoint will not occur when the instruction is executed.

An example may help:

```
000004D8  MOVE.L (0008,A6),D7
000004DC  TST.L D7
000004DE  BLT 00000502
000004E0  MOVEQ #03,D0
000004E2  CMP.L D0,D7
000004E4  BGE 00000502
000004E6  MOVE.L D7,D0
000004E8  LSL.L #4,D0
```

In the above example, the instructions at locations 4DE, 4E2, and 4E6 are 16-bit instructions that are 16-bit aligned. If the user attempts to set a breakpoint at one of these addresses and they have indicated that the address lies in 8 or 16-bit ROM, SourceGate II will round the address up and set a breakpoint at 4E0, 4E4, or 4E8. In the case of the instructions at 4E2 and 4E6, this is not a big problem because the breakpoint will occur just one instruction later. However, in the case of the branch instruction at 4DE, SourceGate II will set the breakpoint at address 4E0 and, if the branch is taken, the breakpoint will never be occur.

The format of the S_ROMWIDTH command is:

S_ROMWIDTH {8 | 16 | 32 | ASK}

Some examples follow:

S_ROMWIDTH 8:
assume 8 or 16-bit ROM for all breakpoints/single-steps

S_ROMWIDTH 16:
assume 8 or 16-bit ROM for all breakpoints/single-steps

S_ROMWIDTH 32:
assume 32-bit ROM for all breakpoints/single-steps

S_ROMWIDTH ASK:
ask the user for the ROM width next time they try to set a breakpoint or single-step in ROM.

2 Changes In Operation From Earlier Versions

Several changes in the operation of SourceGate II have occurred in this release. This chapter will outline these changes.

Single-Step Window No Longer Supported

The Single-Step window has been removed from this version of SourceGate II. If the user wishes to single-step, they should use the Command, Code, or VASM windows.

File/Change Directory Function No Longer Supported

The File/Change Directory function has been removed from this version of SourceGate II. Any functions that deal with files now allow the user to specify the path of the file. Therefore, the File/Change Directory function was no longer needed.

Changes In Watch/ Watch In Real Time Window Operation

The Watch and Watch In Real Time windows have undergone several changes in the way they operate. A complete description of the operation of the Watch and Watch In Real Time windows is given in the following sections.

The Watch and Watch In Real Time windows allow certain memory locations or variables to be "watched" and/or modified by the user. The Watch window allows the user to monitor and/or modify variables while single-stepping or any other time when emulation is halted. The Watch In Real Time window is used to monitor variables in real-time while emulation is in progress. The operation of both of these watch windows is identical.

The Watch window can be displayed by choosing "Display/Watch". The Watch In Real Time window can be displayed by choosing "Display/Watch In Real Time".

Data structure information along with the display of local variables is supported when Archimedes, Software Development Systems (SDS), or any other compiler that can produce IEEE-695, Common Object File Format (COFF), or ELF/DWARF output files is used. Watch windows can be saved to disk and loaded again at a later date. All variable types are supported and can be displayed.

Once a watch variable is being displayed in the Watch or Watch In Real Time window, you can click on the variable line to see more detailed information about the watch variable. For example, if a data structure is being shown, clicking on the data structure watch variable will open an additional window that only contains information about that data structure. If you continue to click on individual elements of the structure, additional windows will be shown for these individual elements and they will automatically be added as watch variables in the Watch window. Arrays are handled in a similar manner so that individual elements of the array can be displayed if so desired.

Adding a Watch Variable

Variables can be added to the Watch window by clicking on the **Add** button or by entering in the variable information manually. The following topics provide additional information on how to add watch variables using each of these methods:

Entering Watch Variables Manually

The Watch window allows the user to enter watch variables manually by clicking on a line in the watch window and typing in the desired type, address, and display type information for the variable to be "watched" in the format of:

type override address display_type

Pressing the **ENTER** key will direct SourceGate II to accept the watch variable as a new entry and its value will then be updated. If the variable to be watched is a pre-defined symbol, the user can bypass the above step by typing the name of the symbol in the watch entry field and pressing the **ENTER** key. SourceGate will search the symbol table information and display the variable in its proper form.

Probably the easiest way to enter a watch variable that is a symbol name is to press the right mouse button while in the watch window. This will invoke the Symbol Selection window from which a symbol can be chosen by clicking on the desired variable and then pressing the **Select** button. Once the symbol has been selected, it will automatically be added to the watch variable list. See the ***Symbol Selection Window*** section of your SourceGate II documentation for additional information.

The following topics provide additional information on entering watch variables manually in this window:

Watching A Symbol

To choose a symbol name as the watch variable, make the Watch window the active window and then press the right mouse button. This will bring up the Symbol Selection window that shows a listing of all available symbols. Click on the desired symbol to watch and then press the **Select** button. The information for the chosen symbol will be automatically loaded in the Watch window. See the ***Symbol Selection Window*** section of your SourceGate II documentation for additional information.

If the user wishes to watch certain elements of a structure, this can be accomplished by entering "structure.element". Similarly, if the user wishes to watch certain elements of an array, this can be accomplished by entering "array_name[element]". For multi-dimensional arrays, type "array_name[a][b]" to watch a particular element. Both structures and arrays will be displayed in separate "Watch Output" windows if the output if too large to be shown in the Watch window.

Watch Variable Type

The type field is used to tell the Watch window what type of variable is being specified. Any type can include an '*' after the type to denote a pointer. Some examples follow:

char Variable at the address is of type char
char* Value is a pointer to a variable of type char
char** Value is a pointer to a pointer that points to a variable of type char

The following types are available and should be entered as shown:

char
short
int
long
unsigned char
unsigned short
unsigned int
unsigned long
float
double
address
byte
word
char*
short*
int*
long*
float*
double*

Watch Variable Overrides
On some emulators, the user can specify an override designator as part of an address. Overrides may precede the address in the form {override}address. For example, if the user wanted to watch an integer in the supervisor/data space of a 68000 based system at location 5000 hex and have it displayed as a decimal value, they should enter "integer {SD}0x5000 %d". For additional information, see ***Watch Entry Override Selection Window.***

Watch Variable Address
The address field is used to tell the Watch window the address of the data to be watched. This field can be entered in decimal (default) or in hexadecimal by preceding the address with "0x" or "0X".

Watch Variable Display Type
The display type field is used by the Watch window to determine the way in which the data will be displayed. The default display type is decimal. All types must be preceded with "%", the case is ignored. An example of the available display types follows:

%c (Character)
%d (Decimal)
%u (Unsigned Decimal)
%f (Float)
%h (or x)(Hex)
%o (Octal)
%b (Binary)
%s (String)

Watch Entry Window
The Watch Entry window allows the user to enter watch variables. From here, the user can enter the type, address, and display type information for the variable to be "watched".

There are two methods that can be used to enter watch variables; manual entry into the Watch Entry window or the usage of the Watch Entry window's buttons. For information on manual entry, see ***Entering Watch Variables Manually***. Information on using the Watch Entry window buttons is provided in the following section.

Watch Entry Window Buttons

The Watch Entry window buttons can be used to enter the information for a watch variable definition. The **Type**, **Symbol**, and **Format** buttons invoke additional windows from which watch variable parameters can be defined.

The buttons available in this window are described below:

Type

The **Type** button allows the user to enter a pre-defined type definition. For more information on selecting a type, see ***Watch Entry Type Selection Window.***

Override

The **Override** button allows the user to select one of the pre-defined overrides. For additional information, see ***Watch Entry Override Selection Window.***

Symbol

The **Symbol** button allows the user to select one of the pre-defined symbols as the variable to be watched. For additional information on selecting a symbol, see the **Symbol Selection Window** section in your SourceGate II documentation.

Format

The **Format** button allows the user to select one of the pre-defined formats definition. For additional information, see ***Watch Entry Format Selection Window.***

Ok

The **Ok** button will enter the defined watch variable into the watch window.

Cancel

The **Cancel** button will cancel the watch variable entry procedure.

Help

The **Help** button will invoke help for this window.

Watch Entry Type Selection Window

The Type Selection window allows the user to choose a pre-defined variable type by using the mouse pointer and left mouse button to highlight the desired type. Pressing the **Ok** button will load the defined type into the Watch Entry window. Additional types can be entered using the manual entry method. These choices represent the more common variable types. The pre-defined choices are as follows:

char
short
int
long
unsigned char
unsigned short
unsigned int
unsigned long
float
double
address
byte
word
char*
short*
int*
long*
float*
double*

Watch Entry Type Selection Window Buttons

The following buttons are available in this window:

Ok

The **Ok** button will load the highlighted type into the Watch Entry window.

Cancel

The **Cancel** button will cancel the type selection and close the Type Selection window.

Help

The **Help** button will invoke help for this window.

Watch Entry Override Selection Window

The Override Selection window allows the user to choose a pre-defined override by using the mouse pointer and left mouse button to highlight the desired override. Pressing the **Ok** button will load the defined override into the Watch Entry window. The overrides shown depend on the BMD being used. See the ***Command Window*** section of your ***SourceGate II Background Mode Debugger User's Guide*** for additional information on the overrides available for your BMD.

Watch Entry Overrides Selection Window Buttons

The following buttons are available in this window:

Ok

The **Ok** button will load the highlighted override into the Watch Entry window.

Cancel

The **Cancel** button will cancel the override selection and close the Type Selection window.

Help

The **Help** button will invoke help for this window.

Watch Entry Format Selection Window

The Format Selection window allows the user to choose a pre-defined variable display format by using the mouse pointer and left mouse button to highlight the desired format. Pressing the **Ok** button will load the defined format into the Watch Entry window. The output value for this watch variable will be in the specified format when displayed in the Watch window. Additional formats can be entered using the manual entry method. These choices represent the more common variable display formats. The pre-defined choices are as follows:

DEFAULT
%c (Character)
%d (Decimal)
%u (Unsigned Decimal)
%f (Float)
%h (or x)(Hex)
%o (Octal)
%b (Binary)
%s (String)

Note that DEFAULT will try and pick out the best display type based on the defined type (e.g. pointers will default to hex, characters to ASCII, integers to decimal, etc.).

Deleting a Watch Variable

Previously entered watch variables can be deleted. To choose the variable to delete, press the left mouse button on the variable to be deleted. This will place the cursor in the line containing the variable. Then, press the left mouse button while pointing to the **Delete** button. The selected watch variable will then be deleted.

Saving a Watch Window

It is possible to save a Watch window for future reference. To save a Watch window, press the left mouse button while pointing to the **Save** button. The user will then be prompted to enter a filename. The Watch window will then be saved as that file.

Restoring a Saved Watch Window

To restore a previously saved Watch window, press the left mouse button while pointing to the **Restore** button. The user will then be prompted to enter the filename of the Watch window to be restored. After choosing the name of the file, the named Watch window will be retrieved and the current values for its variables will be displayed.

Closing a Watch Window

To close a Watch window, the user can either click on the upper left hand corner of the Watch window, or press the left mouse button while pointing to the **Close** button.

Modifying Watch Window Variable Values

The user can modify the values of certain variables in a Watch window. All variable types can be modified with the exception of bit field and floating point variables. To modify a variable's value, simply highlight the value and then type in the new value. The variable's value will immediately be changed in the corresponding memory location.

Watch Window Buttons

The following buttons are available in the Watch window:

Add
Add a variable to the Watch window.

Delete
Remove the highlighted variable from the Watch window.

Save
Save the current Watch window to a file.

Restore
Restore a previously saved Watch window.

Close
Close the current Watch window.

Help
Invoke Watch window help.

Changes In Memory Window Operation

The Memory window has undergone several changes in the way it operates. A complete description of the operation of the Memory window is given in the following sections.

The memory window allows the user to display and change individual memory locations. This window is updated any time the BMD is single-stepped and when emulation is stopped.

To change a memory value, place the cursor on the value to be changed and press the left mouse button. To select the entire number, press the left mouse button and then drag until the whole number is highlighted. A new value can then be entered.

Using the arrow or tab keys, the user can move to the adjacent memory location.

An description of Memory window operation is given in the following sections:

Memory Window Buttons

The following buttons are available in the Memory window:

Compare

The **Compare** button allows the user to compare two blocks of memory to see how they differ in content. For additional information, see ***Memory Compare Window***.

Fill

The **Fill** button allows the user to fill a range of memory locations with a constant value. For additional information, see ***Memory Fill Window***.

Goto

The **Goto** button allows the user to move the base address of the memory display. For additional information, see ***Memory Goto Window***.

I/O

The **I/O** button allows the user to read or write a particular memory location. This can be very useful when testing memory mapped I/O locations. **NOTE:** This command is not available for all BMDs. If this button is grayed, your BMD does not support it. For additional information, see ***Memory I/O Window***.

Move

The **Move** button allows the user to move a block of memory to another location. For additional information, see the ***Memory Move Window***.

Find

The **Find** button allows the user to search a block of memory for a particular data value. For additional information, see ***Memory Search Window***.

Find Next

The **Find Next** button allows the user to perform the search operation again, using the search value previously entered with the **Find** button. For additional information, see ***Memory Search Window***.

Test

The **Test** button allows the user to perform a non-destructive memory test on any RAM memory location. For additional information, see ***Memory Test Window***.

Mode

The **Mode** button allows the user to choose the display format. **NOTE:** This command is not available for all BMDs. If this button is grayed, your BMD does not support it. For additional information, see ***Memory Display Format Window***.

Close

The **Close** button closes the Memory window.

Help

The **Help** button invokes help for the Memory window.

Memory Compare Window

The Memory Compare window is invoked when the **Compare** button is pressed from within the Memory window. This window allows the user to compare two memory blocks to see any differences between them.

The 'Start' field defines the starting address of the first block, the 'End' field defines the ending address of the first block, and the 'Dest' field defines the beginning address of the block to be compared.

The results of this operation are shown in the Memory Output Window. For additional information, see ***Memory Compare Window Buttons*** and ***Memory Output Window***.

Memory Compare Window Buttons

The following buttons are available in this window:

Ok

The **Ok** button will begin the compare operation.

Cancel

The **Cancel** button will cancel the compare operation and close the Memory Compare window.

Help

The **Help** button will provide help for this window.

Memory Fill Window

The Memory Fill window is invoked when the **Fill** button is pressed from within the Memory window. This window allows the user to fill memory with a constant value.

The 'From' field defines the starting address, the 'To' field defines the ending address, and the 'Fill Value' field defines the value to be loaded into this range of memory addresses.

For example, if 'From' contained 5000H, 'To' contained 6000H, and 'Fill Value' contained 33H, every location from 5000H to 6000H would contain 33H upon execution of this command.

For an explanation of the buttons shown in this window, see ***Memory Fill Window Buttons***.

Memory Fill Window Buttons

The following buttons are available in this window:

Ok

The **Ok** button will begin the fill operation.

Cancel

The **Cancel** button will cancel the fill operation and close the Memory Fill window.

Help

The **Help** button will provide help for this window.

Memory Goto Window

The Memory Goto window is invoked when the **Goto** button is pressed from within the Memory window. This window allows the user to change the base address shown in the Memory window.

The 'Address' field is the desired base address which will be shown in the Memory window.

For an explanation of the buttons shown in this window, see ***Memory Goto Window Buttons***.

Memory Goto Window Buttons

The following buttons are available in this window:

Ok

The **Ok** button will make the base address shown in the Memory window to be the address entered.

Cancel

The **Cancel** button will cancel the goto operation and close the Memory Goto window.

Help

The **Help** button will provide help for this window.

Memory I/O Window

The Memory I/O window is invoked when the **I/O** button is pressed from within the Memory window. This window allows the user to either read or write memory mapped I/O locations.

The 'Address' field defines the I/O address where the read or write operation will be performed. The 'Value' field defines the value to be written to the defined address if 'Write' is selected. The 'Value' field will show the contents of the I/O location if 'Read' is selected.

For additional information, see ***Memory I/O Window Buttons***.

Memory I/O Window Buttons

The following buttons are available in this window:

Read

The **Read** button will read the defined I/O location and display the results in the 'Value' field. This button is only shown when the "Read" radio button is selected.

Write

The **Write** button will write the defined I/O location with the value located in the 'Value' field. This button is only shown when the "Write" radio button is selected.

Cancel

The **Cancel** button will cancel the I/O operation and close the Memory I/O window.

Help

The **Help** button will provide help for this window.

Memory Move Window

The Memory Move window is invoked when the **Move** button is pressed from within the Memory window. This window allows the user to move a memory block to another location in the memory map.

The 'Start' field defines the starting address of the block to be moved, the 'End' field defines the ending address of the block to be moved, and the 'Destination' field defines the address where the beginning of the block will be loaded.

For example, if 'Start' contained 5000H, 'End' contained 6000H, and 'Destination' contained 7000H, every location from 5000H to 6000H would be moved to its corresponding location beginning with address 7000H.

For an explanation of the buttons shown in this window, see ***Memory Move Window Buttons***.

Memory Move Window Buttons

The following buttons are available in this window:

Ok

The **Ok** button will begin the move operation.

Cancel

The **Cancel** button will cancel the move operation and close the Memory Move window.

Help

The **Help** button will provide help for this window.

Memory Search Window

The Memory Search window is invoked when the **Find** button is pressed from within the Memory window. This window allows the user to search memory for a particular value. The user can also search for when a value is 'NOT' in memory. This can be a handy feature when trying to verify that memory was initialized correctly.

The 'From' field defines the address to begin the search from, the 'To' field defines the address to search to, and the 'Search Value' field defines the value to be searched for. The 'Search For NOT Match' box, when checked, will search memory and display those locations that do NOT contain the value contained in the 'Value' field.

When a match to the search is found, the matching location is highlighted in the Memory window. If no match is found, the message "Value not found" will be displayed.

For an explanation of the buttons shown in this window, see ***Memory Search Window Buttons***.

Memory Search Window Buttons

The following buttons are available in this window:

Ok

The **Ok** button will begin the search operation.

Cancel

The **Cancel** button will cancel the search operation and close the Memory Search window.

Help

The **Help** button will provide help for this window.

Memory Test Window

The Memory Test window is invoked when the **Test** button is pressed from within the Memory window. This window allows the user to perform a non-destructive memory test on any RAM location.

The 'From' field defines the first memory location to be tested, and the 'To' field defines the last memory location to be tested. The 'Test Type' field allows the user to define the type of test being performed. The 'Short(MT)' test writes an alternating pattern of AA (hex) and 55 (hex) and the 'Long(MTL)' test writes a 'Walking 1' pattern to each location. The test reads back the values that were written to each memory location in the defined range and checks to make sure that the memory is working properly.

The results of this operation are shown in the Memory Output Window. For additional information, see ***Memory Test Window Buttons*** and ***Memory Output Window***.

Memory Test Window Buttons

The following buttons are available in this window:

Ok

The **Ok** button will begin the test operation.

Cancel

The **Cancel** button will cancel the test operation and close the Memory Test window.

Help

The **Help** button will provide help for this window.

Memory Display Format Window

The Memory Display Format window is invoked when the **Mode** button is pressed from within the Memory window. This window allows the user to change the display format of the Memory window.

To select a display format, the user should place the mouse pointer on the desired format and then press the left mouse button. Once this is done and the OK button is pressed, the Memory window will show memory values in the selected display format.

For an explanation of the buttons shown in this window, see ***Memory Display Format Window Buttons***.

Memory Display Format Window Buttons

The following buttons are available in this window:

Ok

The **Ok** button will re-display the Memory window in the chosen display format.

Cancel

The **Cancel** button will cancel the format change operation and close the Memory Display Format window.

Help

The **Help** button will provide help for this window.

Memory Output Window

The Memory Output window shows the result of the previous operation. If the previous operation was a **Compare** operation, the output is of the format:

```
Addr1  Val1  Addr2  Val2
```

where:

Addr1 is the first address in block 1 which differs
Val1 is the value of the block 1 location
Addr2 is the corresponding address in block 2 which differs
Val2 is the value of the block 2 location

If the previous operation was a TEST operation, the output is of the format:

```
Addr  Val1  Val2
```

where:

Addr is the address where the memory test failed
Val1 is the value that was written to memory
Val2 is the bad value that was read from memory

Note that a "WORKING AT ADDRESS ----> XXXXXXXX" message may be displayed in certain instances.

For information on the buttons available from this window, see ***Memory Output Window Buttons***.

Memory Output Window Buttons

The following buttons are available in this window:

Abort

The **Abort** button will abort the current operation.

Close

The **Close** button will close this window.

Help

The **Help** button will provide help for this window.

3 General Information

Several problems have been fixed in this version of SourceGate II. Also, there are some known problems that are platform dependent. This chapter provides information on these issues.

Problem Fixes

The following known problems in v1.2.0 have been fixed in v2.0.0 of SourceGate II:

1) The memory window now highlights correctly on search functions

2) The memory window location is now saved properly on all CLOSE events

3) The memory window controls should now enable and disable properly

4) Memory window speedups have been installed

5) The printing "percent complete" bar now behaves more properly

6) The symbol table type size table has been increased

7) Some IEEE reader errors have been cleaned up

8) The "Print to File or Printer" dialog has been removed on the UNIX version

9) SourceGate Window titles can be set in the .INI file

10) A ^C handler has been added to the UNIX version for more graceful exits

11) The VASM and Tracking Assembly windows now show symbols

12) Unknown IEEE reader errors no longer say "Out of Memory"

13) The last Download (and Upload) directory is remembered between SourceGate sessions

14) The ':' command has been updated with the current list of Storage Classes, and shows the register from the Register Map for symbols that are register variables

15) Changed the "Go From/To ..." dialog to allow symbol entry as well as line number entry.

16) Fixed the "Go From/To ..." dialog line number selection format to match the rest of SourceGate

17) Fixed lockup that occurred when a "Symbols Only" download was selected with an Extended TEKHEX file

18) The Watch Window %d and %u formatting overrides now work as expected.

19) The codeview module list and the list of modules in the symbol selection dialog are now sorted alphabetically by name.

20) New default icons have been installed in the Windows executable.

21) Fixed "Reversed COFF array dimensions" problem

22) The Windows 95 Right Mouse Button conflict has been resolved.

23) The Watch Window would say "Internal Member Error" for certain members of a structure. This was caused by the IAR "attributes" type, and has been fixed.

24) The COFF reader no longer discards bitfields.

25) The Watch Window now displays bitfields (but does not let you edit/change their values).

26) ELF/DWARF structure and union sizes are now handled properly.

27) Fixed Code window button spacing.

28) Fixed problem in certain windows where the text was cut off at the bottom.

29) Changed an algorithm that allows large IEEE files to be read approximately twice as fast as before. This change also sped up reads of other formats by 1-6%.

Known Problems

Every effort has been made to insure that SourceGate II is a solid, problem free interface. However, there are a few known problems with this release. Every effort will be made to fix these problems in a future release.

Additional information is available in the following sections.

Known Problems Affecting All Platforms

The following are problems that have been observed on all platforms:

Source Code Interface Options

Changing options in the Source Code Interface dialog (byte order, etc.) will not immediately update any open windows which are affected by the change. The change for a window will take effect the next time the window is updated with new information.

For example, if the byte order is changed and the Watch window is on-screen, the new byte order will not effect the variables displayed in the Watch window until a Watch window update occurs.

Double And Long Double Values In The Watch Window

If the value of double or long double variables in the Watch window are not correct, the most probable cause is a reversed byte order in the Source Code Interface dialog. Go to the Source Code Interface dialog (choose 'Config/Source Code Interface' from the main SourceGate II menu), and reverse the byte order strings for the Double and Long Double entries. Dismiss the dialog and on the next Watch window update the values for the double and long double variables should be correct.

Known Problems Affecting The Windows Platform

The following are problems that have been observed on the Windows platform:

INI File

Whenever you exit from SourceGate II, the windows that are open, along with their sizes, positions on screen, and other relevant information, are saved in a file named "<BMDName>.ini", where <BMDName> is the name of the simulator that you are using (e.g. "bmd332.ini", "bmd360.ini", etc.). When SourceGate II is restarted later, it will reference this file and reopen all of the windows that were active when you last exited, placing them at their previous locations on screen. Under normal circumstances this works well without any problems. However, should you change your Windows screen resolution, your SourceGate II windows may be reopened at odd locations and/or with unusable sizes. As a precaution, whenever you change your Windows screen resolution you should delete the SourceGate II "<BMDName>.ini" file.

Note, however, that deleting the SourceGate II "<BMDName>.ini" file will also destroy any saved preferences settings.

Known Problems Affecting The SUN/SPARC Platform

The following are problems that have been observed on the SUN/SPARC platform:

Printing

In order to print, you must first print to a file. You may then print the file to your printer. Printing directly from SourceGate II to a printer is not reliable and may fail with an "Internal XVT Error".

Iconizing A Window

If a window is iconized, the only way to reopen the iconized window is by clicking on the icon. The window cannot be reopened by selecting it from the open windows list.

Monochrome Monitors

On systems using monochrome monitors, some windows will display a black cursor on a black background, rendering the cursor difficult to see. This occurs most notably in the "Config/Memory" window.

Help and Modal Dialogs

The menu, buttons, and cross-references in the Help window do not work if Help is brought up over a Modal dialog (such as the Event Address Binary or Watch Entry dialogs). The modal dialog behind the Help window must be dismissed before the Help window controls will work.

Num Lock

On some systems, when "Num Lock" is lit (selected), the space bar no longer works in any text edit box. This problem has been noted on a SPARCStation IPX with a Type 5 keyboard and on a SPARCStation IPC with a Type 4 keyboard, both running SunOS 4.1.3. It has not been a problem on a SPARCStation LX running SunOS 5.3 with a Type 5 keyboard.

Warnings out of 'ld.so'

SourceGate II for SUN was compiled on a Sun SPARCStation running SunOS 4.1.3_U1. If your version of SunOS does not match 4.1.3_U1, then you may receive the following warning message each time SourceGate II is started:

```
ld.so: warning:/usr/lib/libc.so.1.x  has older revision than expected x
```

This warning message is generated because your version of the shared library 'libc' does not match the version expected by SourceGate II. It has been our experience that this message is annoying but harmless. The functions used by SourceGate II from this shared library are unchanged from previous versions of libc and, therefore, older versions of libc may be safely used by SourceGate II.

Known Problems Affecting The HP Platform

The following are problems that have been observed on the HP platform:

Printing

In order to print, you must first print to a file. You may then print the file to your printer. Printing directly from SourceGate II to a printer is not reliable and may fail with an "Internal XVT Error".

Iconizing A Window

If a window is iconized, the only way to reopen the iconized window is by clicking on the icon. The window cannot be reopened by selecting it from the open windows list.

Monochrome Monitors

On systems using monochrome monitors, some windows will display a black cursor on a black background, rendering the cursor difficult to see. This occurs most notably in the "Config/Memory" window.

Help and Modal Dialogs

The menu, buttons, and cross-references in the Help window do not work if Help is brought up over a Modal dialog (such as the Event Address Binary or Watch Entry dialogs). The modal dialog behind the Help window must be dismissed before the Help window controls will work.

Menu Bar Resizing

When SourceGate II is started, the main menu bar will shrink and resize itself several times. This behavior is part of the normal startup sequence and is not a bug.

Index

C

C++ Class Browser, 1, 2
C++ Class Sub-Browser, 1, 3

E

ELF/DWARF, 1

F

File Download Speed Ups, 1

G

General Information, 31

K

Known Problems, 33

M

Memory Window, 1, 22

N

New BMD Commands, 7
New Features, 1

P

Preferences, 1, 4
Problem Fixes, 31

R

Register Map, 1, 5

S

S_PMBA Command, 1, 7
S_ROMWIDTH Command, 1, 8

T

Tracking Assembly, 1, 6

V

VASM, 1, 6

W

Watch/Watch In Real Time Window, 13
Windows 95, 1